

**Amendments to the Claims:**

This listing of claims will replace all prior versions, and listings, of claims in the application:

**Listing of Claims:**

1. (Currently Amended) A linking method of for forming an executable program from a plurality of object code modules, each object code module comprising section data and an ordered sequence of relaxation instructions including a first type defining relocation operations and a second type controlling linker operations, each relaxation instruction having associated therewith a unique instruction count of a sequence of instruction counts determining the ordered sequence of relaxation instructions, wherein the second type includes a jump relaxation instruction which specifies the instruction count of a relaxation instruction to be subsequently read, the linking method comprising:

(a) reading each relaxation instruction in the ordered sequence defined by the sequence of instruction counts; and

(b) where said relaxation instruction is of the first type defining a relocation operation, executing that relocation operation on section data to which it relates, and

where said relaxation instruction is a jump relaxation instruction, a next relaxation instruction which is read is that of the instruction count specified in the jump relaxation instruction;

wherein the formed executable program comprises section data from the plurality of object code modules, relocated in accordance with the executed relaxation instructions.

2. (Previously Presented) A method according to claim 1, further comprising recording a pass value indicative of a number of times said ordered sequence of relaxation instructions has been repeated.

3. (Previously Presented) A method according to claim 2, further comprising detecting when a result of a relocation operation executed on said section data has changed

between repetitions of said ordered sequence of relaxation instructions and recording a change value indicative of an occurrence of said change.

4. (Original) A method according to claim 1 wherein said first type of relaxation instruction specifies an offset and number of bytes of section data to be copied to said executable program.

5. (Original) A method according to claim 1 wherein said first type of relaxation instruction specifies a byte of section data to be copied to said executable program.

6. (Original) A method according to claim 1 wherein said relaxation instruction of the second type includes a conditional relaxation instruction which specifies a number of subsequent relaxation instructions to be skipped in response to a condition being met.

7. (Original) A method according to claim 1 wherein said relocation instruction to be executed specifies a number of subsequent relaxation instructions which are to be repeatedly subsequently read until a condition is met.

8. (Previously Presented) A method according to claim 6 wherein said executing comprises accessing a stack.

9. (Original) A method according to claim 8 wherein said condition is determined according to a value of a top of said stack.

10. (Currently Amended) A computer-readable medium having a computer program stored thereon to provide a A-linker for preparing an executable program from a plurality of object code modules, each object code module comprising section data and an ordered sequence of relaxation instructions including a first type defining relocation operations and a second type controlling linker operations, each relaxation instruction having associated

therewith a unique instruction count of a sequence of instruction counts determining the ordered sequence of relaxation instructions, the linker comprising:

a relaxation module for reading the relaxation instructions in an ordered sequence defined by the sequence of instruction counts, and for executing said relaxation instruction; and

a section data module for holding section data and outputting the prepared executable program;

wherein, when said relaxation instruction defines a relocation operation, the relaxation module executes said relocation operation on section data defined in one or more previous relaxation instructions and, when said relaxation instruction identifies section data, said section data is copied to said ~~executable program~~ section data module, said section data being relocatable by subsequent previous relocation operations.

11. (Currently Amended) A computer-readable medium linker according to claim 10 wherein said linker comprises, ~~further comprising~~ a state flag means for holding a pass value indicative of a number of times the ordered sequence of said relaxation instructions has been repeated.

12. (Currently Amended) A computer-readable medium method according to claim 11 wherein said linker comprises state flag means for recording a change value to indicate that copied section data has changed between repetitions of execution of a same relaxation instruction of the first type.

13. (Currently Amended) A computer-readable medium linker according to claim 10 wherein the relaxation module comprises means for deriving an offset and a number of bytes of section data to be copied to said executable program from said second type of relaxation instruction.

14. (Currently Amended) A computer-readable medium linker according to claim 10 wherein the relaxation module comprises means for defining a byte of section data to be copied to said executable program.

15. (Currently Amended) A computer-readable medium ~~linker~~ according to claim 10 wherein the linker comprises a stack capable of holding values generated by execution of the relaxation instructions.

16. (Currently Amended) A computer-readable medium ~~linker~~ according to claim 15 wherein a value of a top of the stack is readable by the relaxation module to determine a condition.

17. (Currently Amended) A computer-readable medium ~~linker~~ according to claim 10, wherein said linker comprises a, ~~further comprising~~ means for determining whether a condition specified in a relaxation instruction of a third type is met, to determine a number of subsequent relaxation instructions to be skipped.

18. (Currently Amended) A computer program product for forming an executable program from a plurality of object code modules, said computer program product comprising program code means having section data and an ordered sequence of relaxation instructions including a first type defining relocation operations and a second type controlling linker operations, each relaxation instruction having associated therewith a unique instruction count of a sequence of instruction counts determining the ordered sequence of relaxation instruction associated therewith, said program code means being arranged so that, when run on a computer, the following are performed:

(a) reading each relaxation instruction in the ordered sequence defined by the sequence of instruction counts; and

(b) where said relaxation instruction is of the first type defining a relocation operation, executing that relocation operation on section data to which it relates, and

where said relaxation instruction is a jump relaxation instruction, a next relaxation instruction which is read is that of the instruction count specified in the jump relaxation instruction;

wherein the formed executable program comprises section data from the plurality of object code modules, relocated in accordance with the executed relaxation instructions.

19. (Currently Amended) A linking method of for forming an executable program from a plurality of object code modules, each object code module comprising section data and an ordered sequence of relaxation instructions including a first type defining relocation operations and a second type controlling linker operations wherein the second type includes a conditional relaxation instruction which determines whether subsequent relaxation instructions are executed depending on a condition defined in the condition relaxation instruction, the linking method comprising:

(a) reading each relaxation instruction of the ordered sequence; and

(b) where said relaxation instruction is of the first type defining a ~~relaxation~~ relocation operation, executing that relocation operation on section data to which it relates, and

where said relaxation instruction is a jump relaxation instruction, a next relaxation instruction which is read is that of an instruction count specified in the jump relaxation instruction,

and where the relaxation instruction is a conditional relaxation instruction, accessing a state variable which denotes linker state to determine whether the condition is satisfied,

wherein the formed executable program comprises section data from the plurality of object code modules, relocated in accordance with the executed relaxation instructions.

20. (Previously Presented) A method according to claim 19 wherein the state variable is a pass value indicative of a number of times said ordered sequence of said relaxation instructions have been repeated.

21. (Previously Presented) A method according to claim 19 wherein the state variable is a change flag denoting whether a symbol has changed value during repeated executions of the ordered sequence of relaxation instructions.

22. (Currently Amended) A method of forming an executable program from a plurality of object code modules, each object code module comprising an ordered sequence of relaxation instructions including a first type defining section data, a second type defining

relocation operations and a third type controlling linker operations, each relaxation instruction having associated therewith a unique instruction count of a sequence of instruction counts determining the ordered sequence of relaxation instructions, the method comprising:

reading each relaxation instruction in the ordered sequence defined by the sequence of instruction counts;

where said relaxation instruction defines a relocation operation, executing said relocation operation on section data defined in one or more previously read relaxation instructions; and

where said relaxation instruction defines section data, copying said section data to said executable program, said section data being relocatable by subsequent previous relocation operations.